ОПЕРАТОРЫ
(продолжение)

Операторы присваивания

- Операторы присваивания являются выполняемыми операторами которые передают значения переменным или элементам массива.

- Существует два основных типа операторов присваивания:
  - **выполняемые**
  - **операторы присвоения значений метки целой переменной**

Выполняемый оператор присваивания.

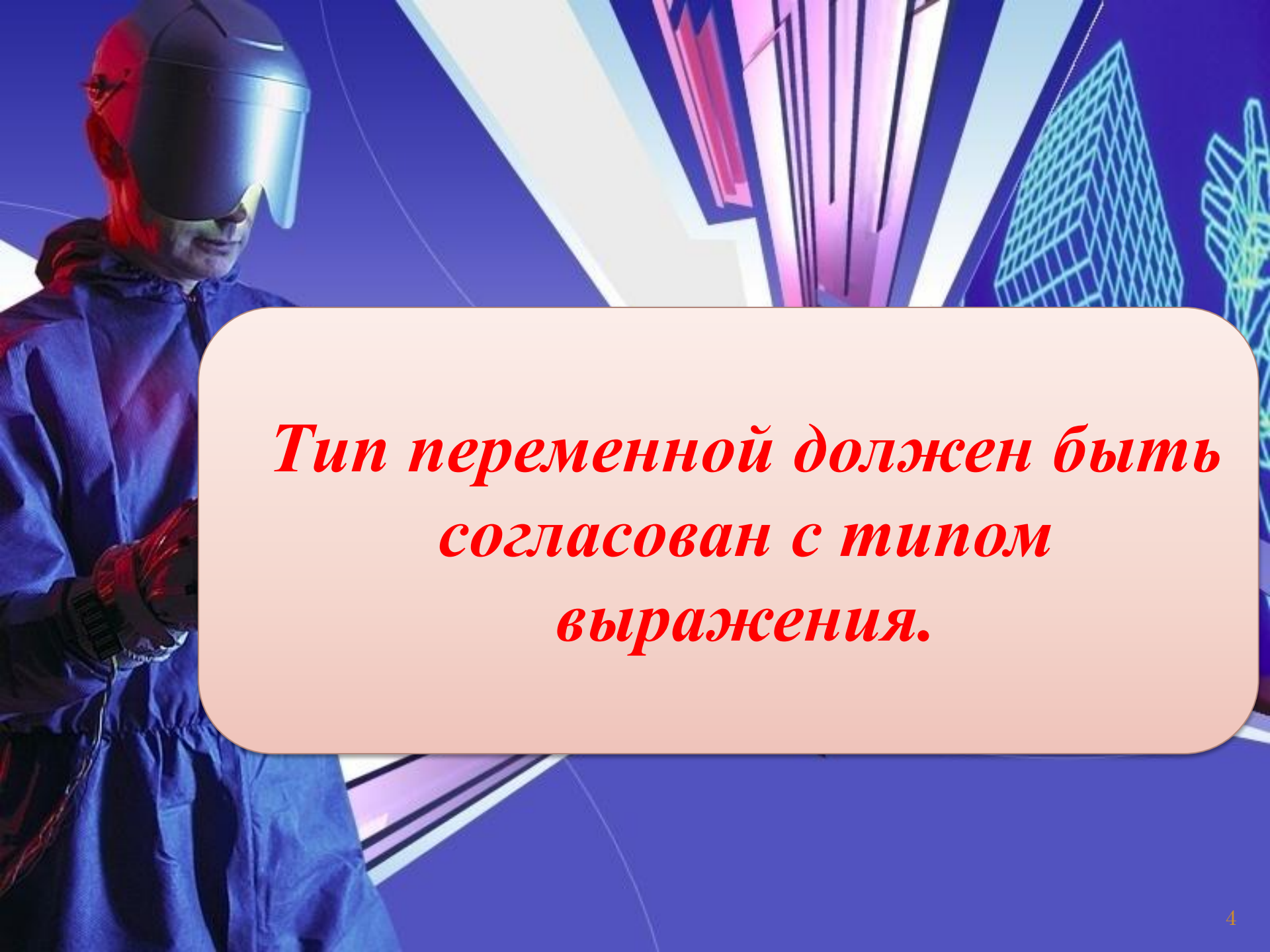
Синтаксис:

Переменная = выражение

Где

переменная - *обычная переменная или элемент массива*

выражение - *любое выражение.*



Тип переменной должен быть согласован с типом выражения.

Если типы элементов арифметического оператора присваивания не совпадают, значения выражения автоматически преобразовываются к типу переменной.

Правила преобразования – самостоятельное изучение!!!

ВЫПОЛНЯЕМЫЙ ОПЕРАТОР ПРИСВАИВАНИЯ

Оператор присвоения значений метки целой переменной.

Синтаксис:

ASSIGN метка TO переменная

Где

метка - *метка формата или метка оператора*

переменная - *целая переменная.*

При выполнении оператора **ASSIGN** переменная принимает значение метки.

Например:

После выполнения оператора

ASSIGN 27 TO K

переменная **K** получает значение метки, равное **27**,

и только после этого она может быть использована в программе,

а именно, в **присваиваемом операторе перехода**.

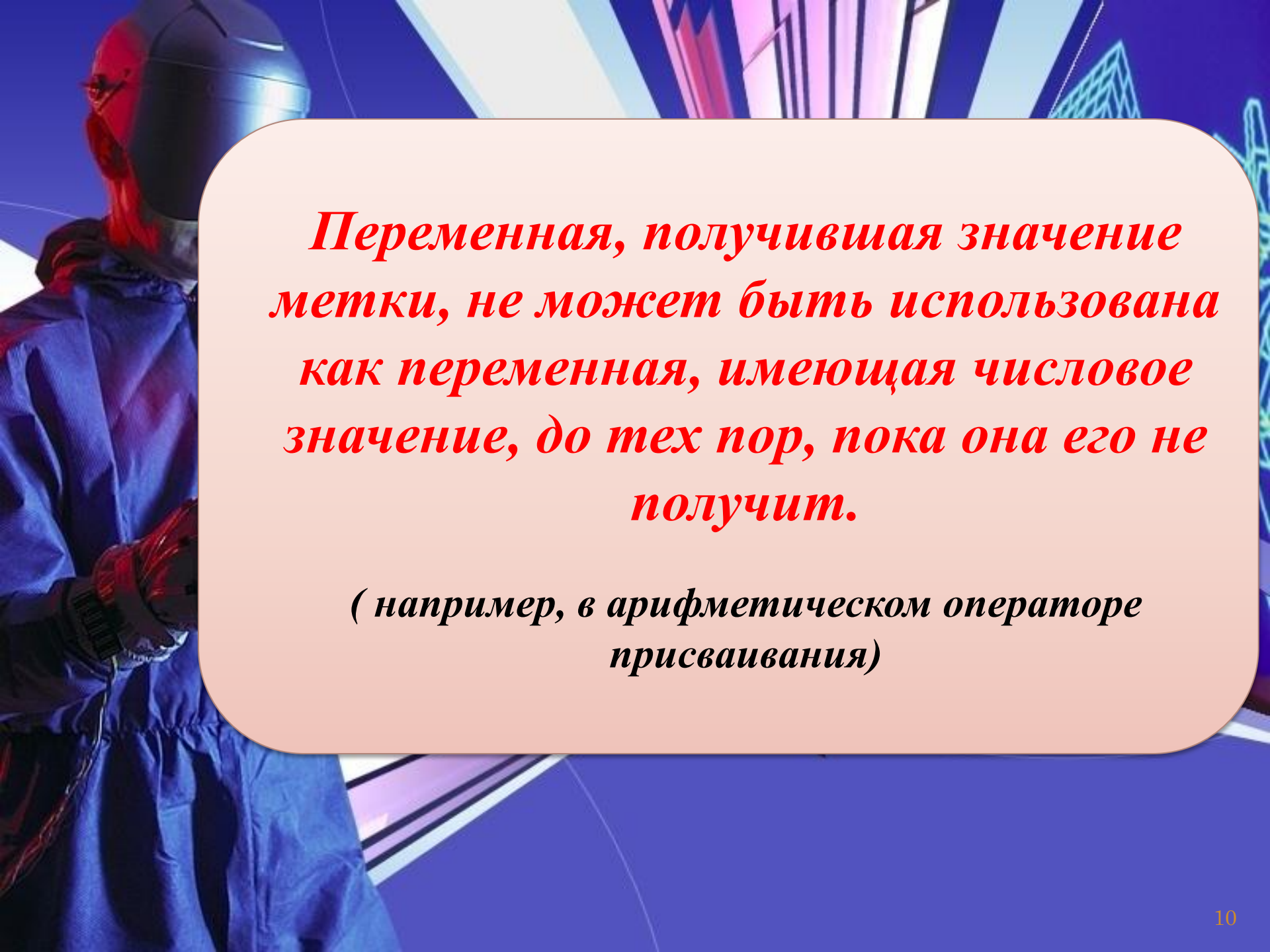
- *Метка должна относиться к любому оператору, который содержится в той же программой единице, что и оператор* **ASSIGN**.

ОПЕРАТОР ПРИСВОЕНИЯ ЗНАЧЕНИЙ МЕТКИ ЦЕЛОЙ ПЕРЕМЕННОЙ

Не следует путать значение метки и значение, которое присваивается переменной.

В следующем примере значение переменной **IVKL** не обязательно должно быть равно **400**:

ASSIGN 400 TO IVKL



Переменная, получившая значение метки, не может быть использована как переменная, имеющая числовое значение, до тех пор, пока она его не получит.

(например, в арифметическом операторе присваивания)

Операторы управления

- Операторы управления устанавливают порядок выполнения операторов Фортрана.

Оператор	НАЗНАЧЕНИЕ
CALL	***Вызывает и выполняет подпрограммы из других программных единиц.***
CONTINUE	***Используется преимущественно для продолжения пути с того места, где стоят намеченные операторы, в частности используется в качестве конечного оператора в операторе цикла* DO.**
DO	***Организует повторное выполнение операторов следующих за* DO *и вплоть до оператора помеченного меткой указанной в операторе* DO.**
END	***Завершает выполнение программой единицы.***
PAUSE	***Останавливает выполнение программы, до тех пор, пока не будет нажата клавиша ENTER.***
RETURN	***Возвращает управление в программную единицу из которой была вызвана данная подпрограмма или функция.***
STOP	***Оператор полного останова, прекращает выполнение программы.***

ОПЕРАТОРЫ УПРАВЛЕНИЯ.

Оператор	НАЗНАЧЕНИЕ
IF	***Организует выполнение по условию других операторов, зависящих от значений определенного выражения.*** ***Различают следующие виды оператора:*** • ***IF - арифметический,*** • ***IF – логический,*** • ***блок.***
ELSE	***Отмечает начало блока*** **ELSE.**
ELSEIF	***Инициирует вычисление выражения.*** **ELSEIF** ***(выражение)*** **THEN.**
ENDIF	***Указывает конец набора серии операторов, следующих за блоком оператора*** **IF.**
GO TO	***Передает управление в какое-либо место программы, включает следующие разновидности:*** • ***присваиваемый,*** • ***вычисляемый,*** • ***безусловный.***

ОПЕРАТОРЫ УПРАВЛЕНИЯ.

Операторы перехода

- Различают **безусловный**, **вычисляемый** и **присваиваемый** операторы перехода.

Безусловный оператор перехода.

Передает управление оператору, помеченному меткой

Синтаксис:

GOTO метка

Где

метка - *метка оператора, выполняемого в той же подпрограмме, что и GOTO*

Пример:

```
C   Пример безусловного GOTO
    GOTO 4022
    . . . . . .
4022  CONTINUE
```

БЕЗУСЛОВНЫЙ ОПЕРАТОР ПЕРЕХОДА

Вычисляемый оператор перехода.

Передает управление оператору, помеченному i-ой меткой в программе

Синтаксис:

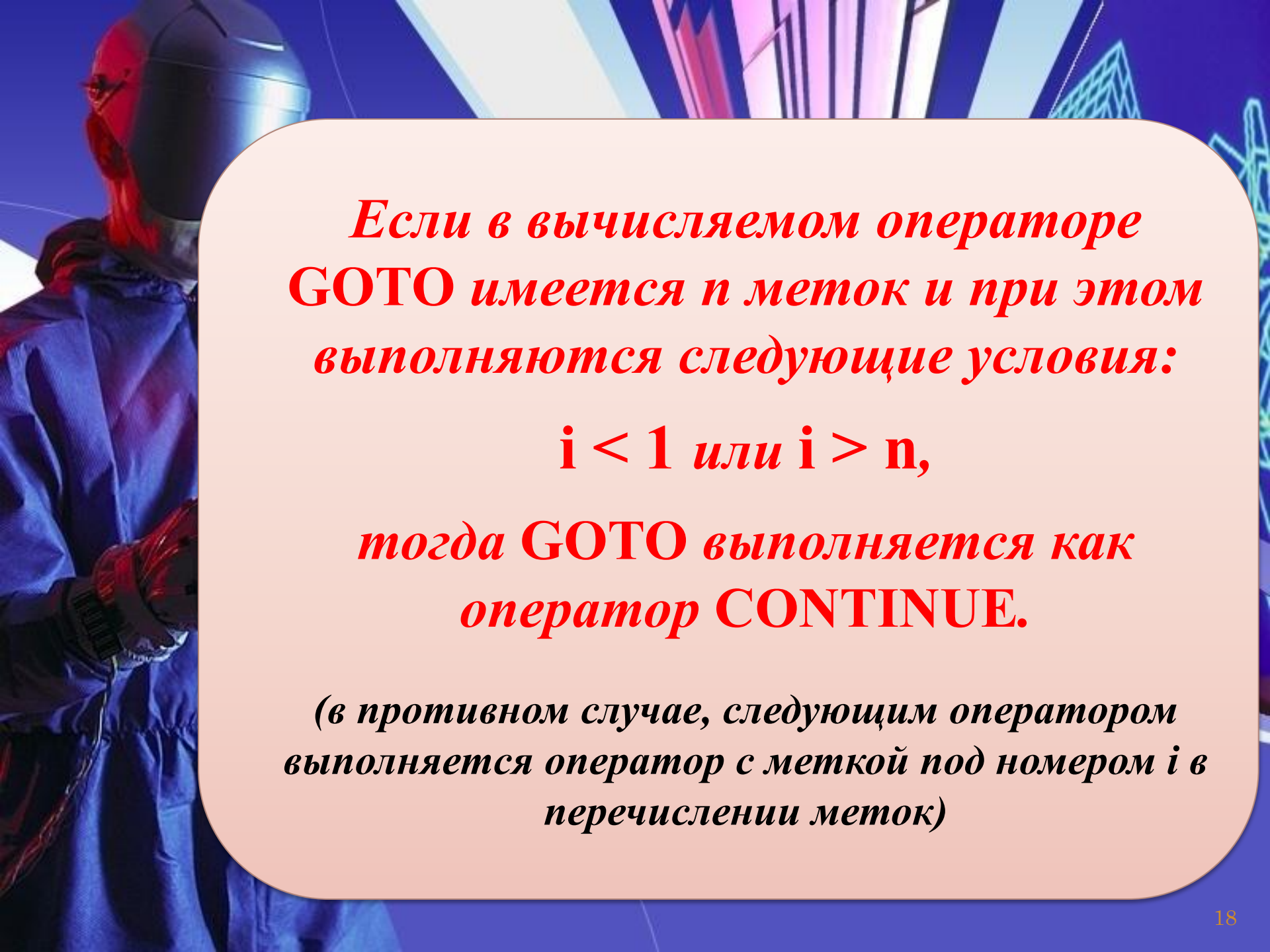
GOTO (метка [,метка]...) [,] i

Где

метка - *метка оператора, выполняемого в той же подпрограмме, что и GOTO*

i - *целое выражение.*

при перечислении меток, одна и та же метка может повторяться



Если в вычисляемом операторе* GOTO *имеется n меток и при этом выполняются следующие условия:

i < 1 ***или*** **i > n,**

***тогда* GOTO *выполняется как оператор* CONTINUE.**

(в противном случае, следующим оператором выполняется оператор с меткой под номером i в перечислении меток)

Пример:

```
C   Пример вычисляемого GOTO
      I = 1
      GOTO (10,20) I
      . . . . . .
10 CONTINUE
      . . . . . .
20 CONTINUE
```

Присваиваемый оператор перехода.

Передает управление на оператор с меткой, чье значение равно значению переменной

Синтаксис:

GOTO имя [[,] (метка [, метка]...)]

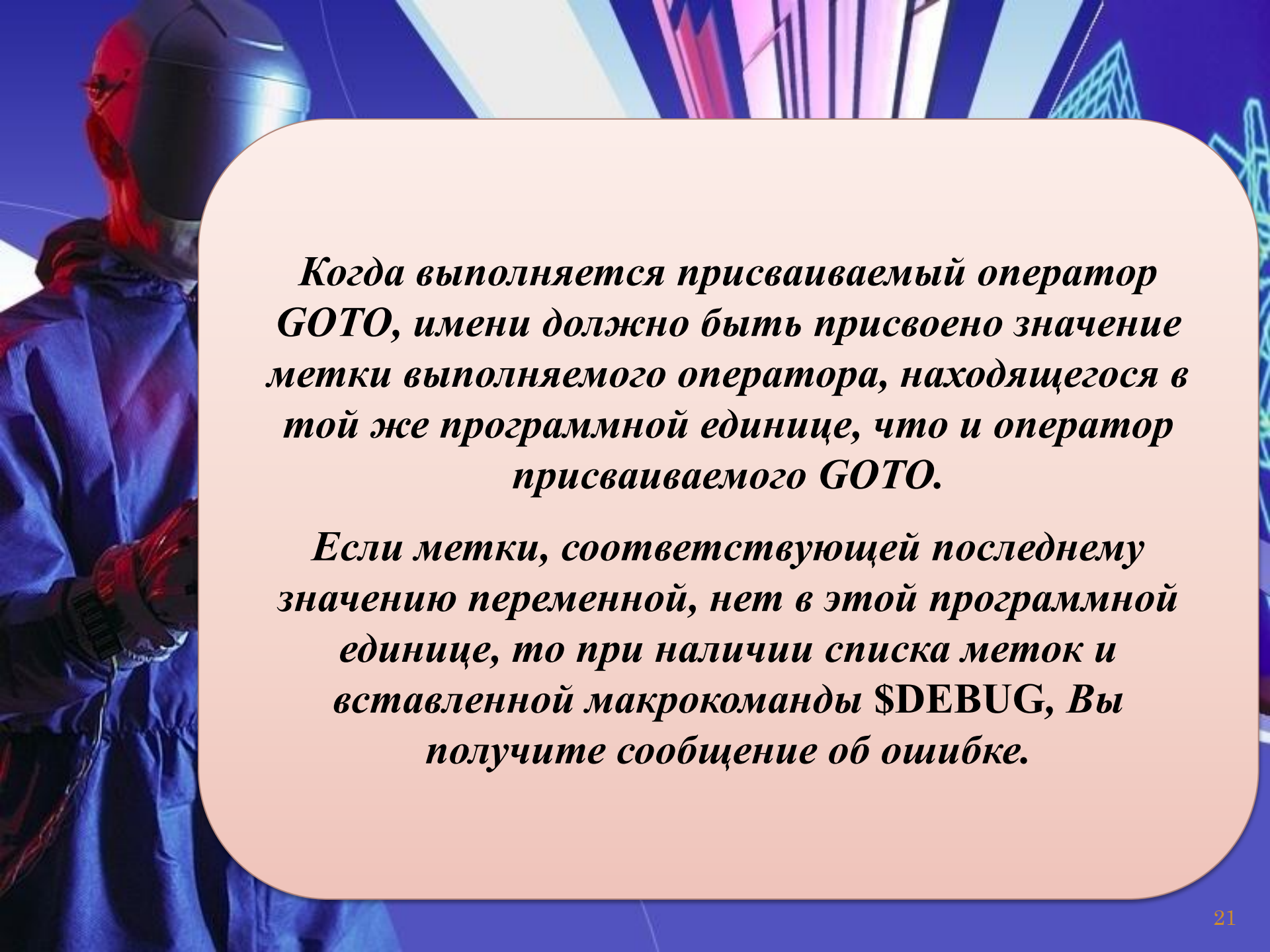
Где

имя - *имя целой переменной значение котрой определено оператором* **ASSIGN.**

метка - *метка оператора, выполняемого в той же подпрограмме, что и GOTO*

при перечислении меток, одна и та же метка может повторяться

ПРИСВАИВАЕМЫЙ ОПЕРАТОР ПЕРЕХОДА



Когда выполняется присваиваемый оператор GOTO, имени должно быть присвоено значение метки выполняемого оператора, находящегося в той же программной единице, что и оператор присваиваемого GOTO.

Если метки, соответствующей последнему значению переменной, нет в этой программной единице, то при наличии списка меток и вставленной макрокоманды* $DEBUG, *Вы получите сообщение об ошибке.

Пример:

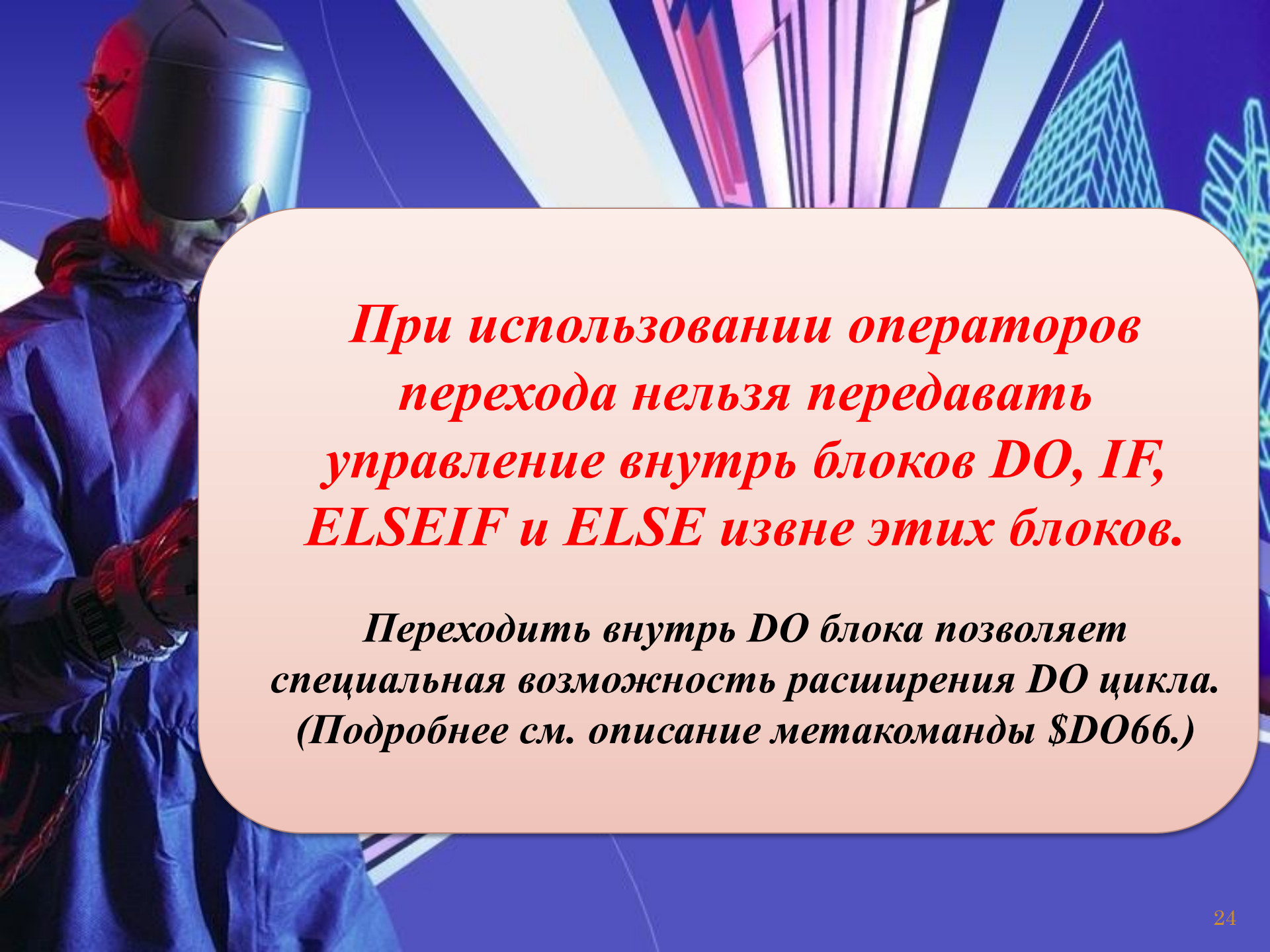
```
C   Пример присваиваемого GOTO
    ASSIGN 10 TO I
    GOTO  I (10,20)
    . . . . . .
10  CONTINUE
    . . . . . .
    GOTO  I
    . . . . . .
20  CONTINUE
```

ПРИСВАИВАЕМЫЙ ОПЕРАТОР ПЕРЕХОДА

Практически вычисляемый и присваиваемый варианты оператора **GO TO** взаимозаменяемы.

Расположение меток в присваиваемом операторе* GO TO *не имеет значения, в то время как в вычисляемом операторе перехода оно является решающим.

ПРИСВАИВАЕМЫЙ ОПЕРАТОР ПЕРЕХОДА



При использовании операторов перехода нельзя передавать управление внутрь блоков DO, IF, ELSEIF и ELSE извне этих блоков.

Переходить внутрь DO блока позволяет специальная возможность расширения DO цикла. (Подробнее см. описание метакоманды $DO66.)

Условные операторы управления

- В Фортране существует три типа условных операторов: **арифметический**, **логический** и **блок**.

- ***При их использовании необходимо помнить о том, что нельзя передавать управление внутрь блоков*** **DO, IF, ELSEIF** ***и*** **ELSE** ***извне этих блоков.***

Арифметический условный оператор.

Вычисляет выражение и передает управление оператору, помеченному одной из описанных меток в соответствии с результатом выражения

Синтаксис:

IF (выражение) метка1, метка2, метка3

Где

выражение - *целое или действительное выражение.*

метки 1,2,3 - *метка операторов, выполняемых в той же подпрограмме, что и оператор* **IF.**

Особенности:

- *Среди трех указанных одна и таже метка может использоваться не один раз.*
- *Первой метке управление передается в случае, если выражение* **<0**, *второй - если* **=0**, *третий - если* **>0** .
- *Следующим после* **IF** *выполняется оператор с указанной меткой.*

АРИФМЕТИЧЕСКИЙ УСЛОВНЫЙ ОПЕРАТОР

Пример:

```
C   Пример арифметического IF
    I = 0
    IF (I) 10, 20, 30
    . . . . . .
10 CONTINUE
    . . . . . .
20 CONTINUE
        . . . . . .
30 CONTINUE
```

Логический условный оператор.

Вычисляется логическое выражение, если его значение* .TRUE.*, то выполняется данный оператор, если выражение* .FALSE.*, то оператор не выполняется, а выполняется следующий за* IF *оператор

Синтаксис:

IF (выражение) оператор

Где

выражение - *логическое выражение.*

оператор - *выполняемый оператор, кроме* **DO**, *блока* **IF**, **ELSEIF**, **ELSE**, **ENDIF**, **END** *и других логических* **IF** *операторов.*

Пример:

```
C   Пример логического IF
    I = 0
    IF (I .EQ. 0) J = 2
    . . . . . .
    IF (J .GT. 2) GOTO 100
    . . . . . .
100   CONTINUE
```

Блок IF THEN .

Вычисляется логическое выражение, если его значение* .TRUE.*, то выполняются операторы, входящие в* IF *блок.

Если выражение* .FALSE.*, то управление передается следующим* ELSE, ELSEIF *или* ENDIF *операторам того же* IF-*уровня.

Синтаксис:

IF (выражение) THEN

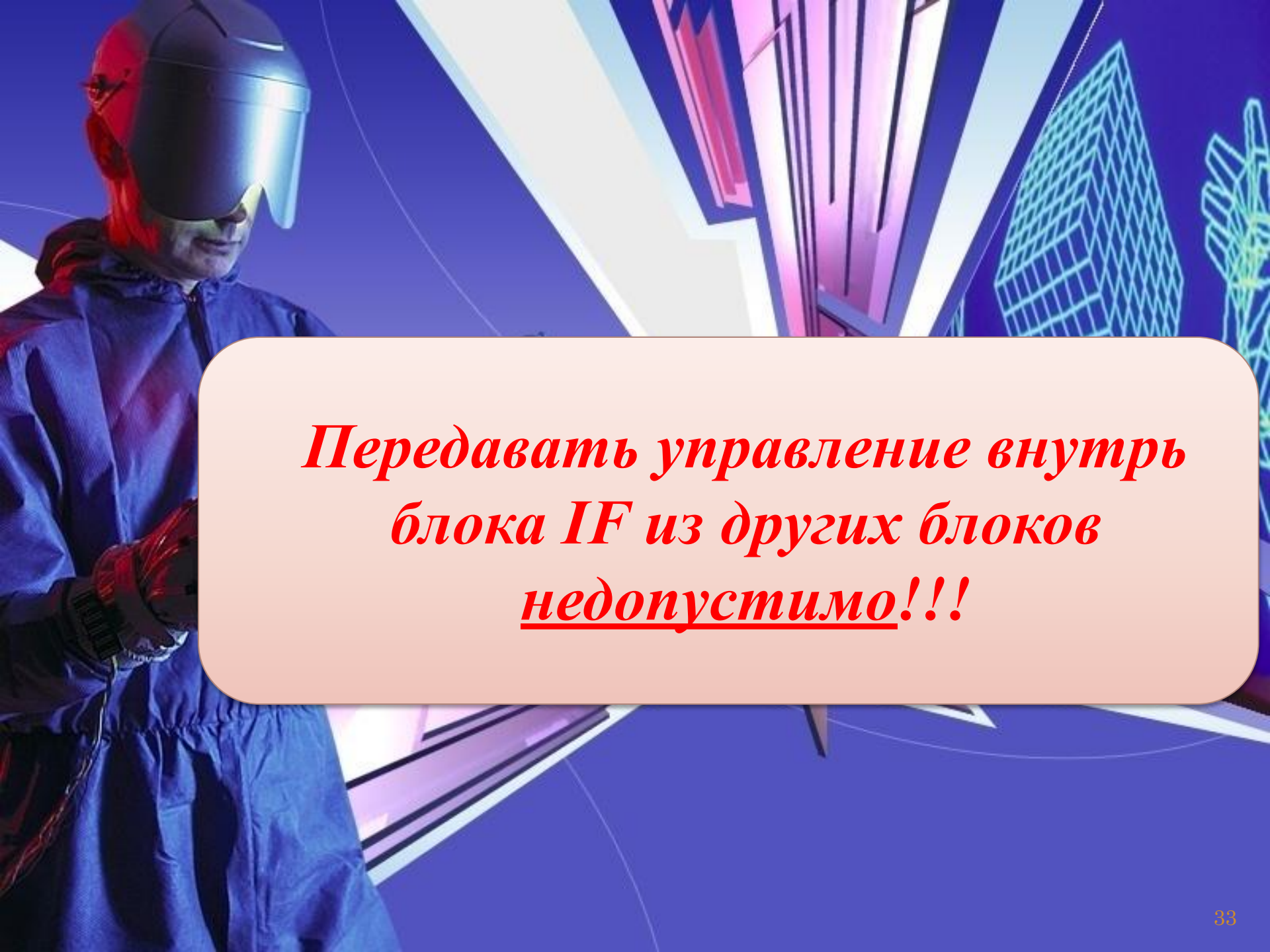
Где

выражение - *логическое выражение.*

Особенности:

- **IF** *блок содержит все выполняемые операторы (возможно ни одного), которые следуют за оператором* **IF** *и до следующего оператора* **ELSEIF**, **ELSE** *или* **ENDIF** *этого же уровня блока* **IF**.
- *После выполнения последнего оператора в блоке* **IF** *выполняется оператор* **ENDIF** *того же уровня.*
- *Если выражение данного блока* **.TRUE.** *и блок не имеет выполняемых операторов, следующим оператором является* **ENDIF** *того же уровня.*
- *Если выражение* **.FALSE.** *то следующим оператором является* **ELSEIF**, **ELSE** *или* **ENDIF** *того же уровня, что и* **IF**.

БЛОК IF THEN



Передавать управление внутрь блока IF из других блоков <u>недопустимо</u>!!!

Пример:

```
C   Пример блока IF THEN
    I = 0
    IF (I .EQ. 0) THEN
    . . . . . .
    ENDIF
    . . . . . .
100   CONTINUE
```

Оператор ELSE.

***Отмечает начало блока* ELSE.**

Выполнение самого оператора не оказывает влияния на программу

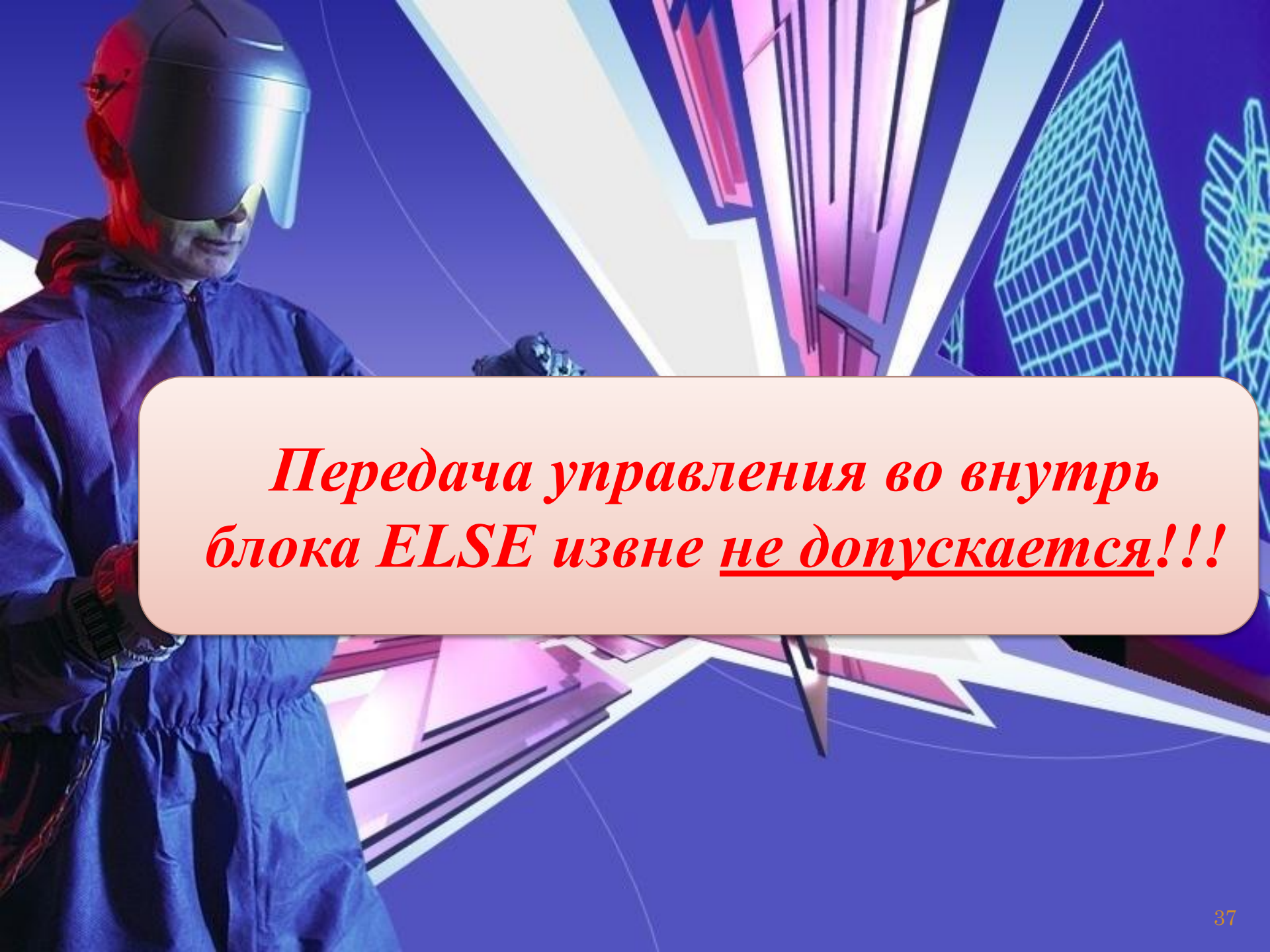
Синтаксис:

ELSE

Особенности:

- *Присоединенный блок* **ELSE** *содержит выполняемые операторы (возможно ни одного), которые следуют за оператором* **ELSE** *и до (но не включая его) оператора* **ENDIF**, *относящегося к тому же оператору* **IF**, *к которому относится* **ELSE**.

- *Соответствующий оператор* **ENDIF** *должен появиться перед любым начальным оператором* **ELSE** *или* **ELSEIF** *другого уровня оператора* **IF**.


Передача управления во внутрь
блока ELSE извне не допускается!!!

Пример:

```
C   Пример блока IF THEN ELSE
    CHARACTER C
    . . . . . .
    READ (*,'(A)') C
    IF (C.EQ.'A') THEN
    CALL  ASUB
    ELSE
    CALL OTHER
    ENDIF
    . . . . . .
```

Оператор ELSEIF.

Синтаксис:

ELSEIF (выражение) THEN

Где

выражение - *логическое выражение.*

Оператор* ELSEIF *инициирует вычисление выражения.

Если значение выражения есть "истина" и в блоке* ELSEIF *присутствует по крайней мере один оператор, то ближайший оператор, который выполняется, является первым оператором* ELSEIF *блока

Особенности:

- *Присоединенный блок* **ELSEIF** *содержит выполняемые операторы (возможно ни одного), которые следуют за оператором* **ELSEIF** *и вплоть до ближайших* **ELSEIF**, **ELSE** *или* **ENDIF** *операторов, которые располагаются на том же уровне оператора* **IF**, *что и данный оператор* **ELSEIF**.

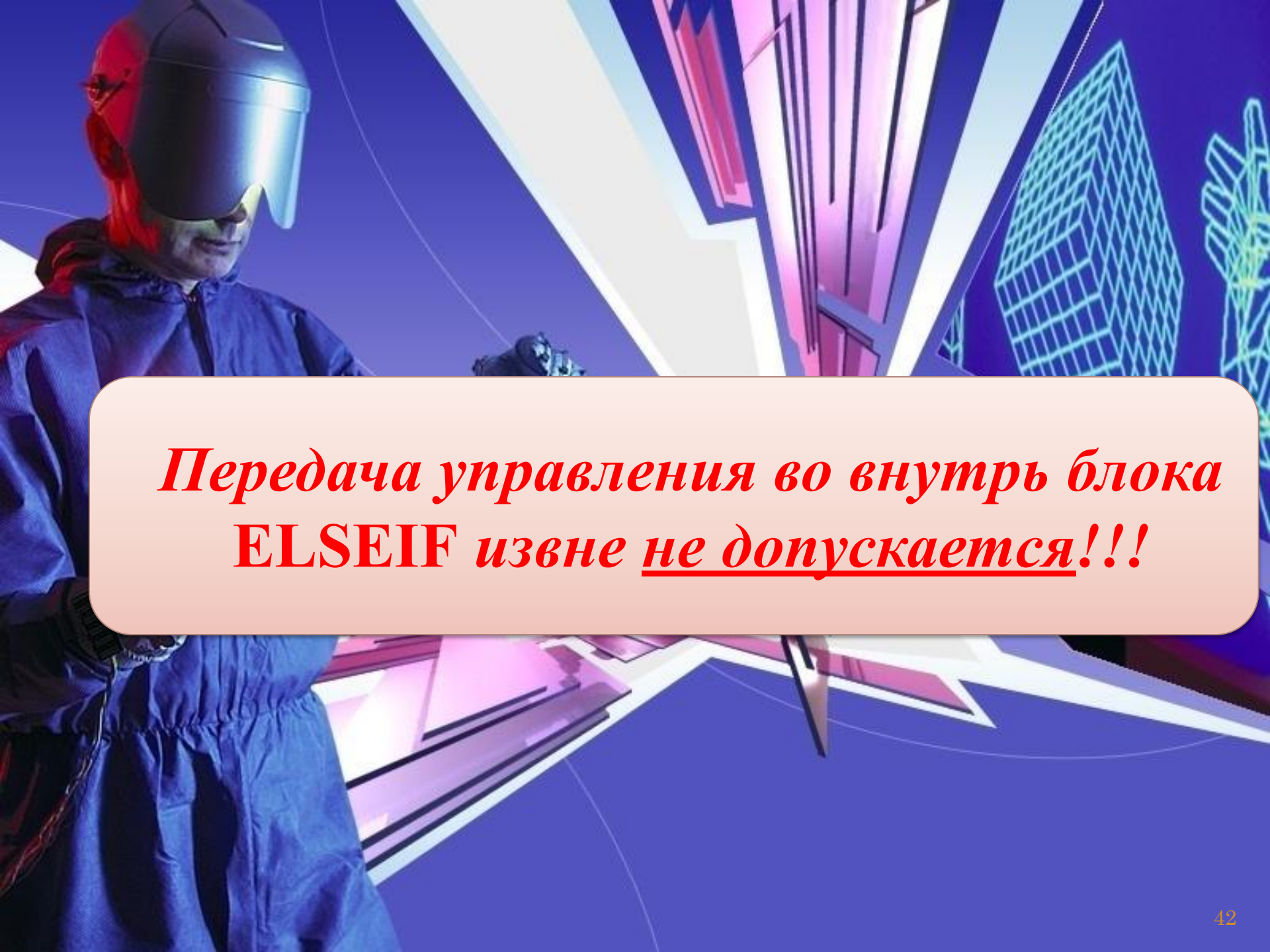
- *После выполнения всех операторов* **ELSEIF**-*блока выполняется оператор, следующий за оператором* **ENDIF** *того же уровня* **IF**, *что и данный оператор* **ELSEIF**.

ОПЕРАТОР ELSEIF

Особенности:

- *Если выражение оператора* **ELSEIF** *оценено как "истина" и блок оператора* **ELSEIF** *не содержит ни одного выполняемого оператора, то ближайшим выполняемым оператором является ближайший оператор* **ENDIF** *того же условного уровня, что и оператор* **ELSEIF**.

- *Если выражение оцененно как "ложь", то следующим выполняемым оператором является ближайший* **ELSEIF**, **ELSE** *или* **ENDIF**, *которые расположенны на том же условном уровне, что и данный* **ELSEIF**.



Передача управления во внутрь блока* ELSEIF *извне <u>не допускается</u>!!!

Пример:

```
C   Пример блока ELSEIF
    CHARACTER C
    . . . . . .
    READ (*,'(X)') C
    IF (C.EQ.'A') THEN
    CALL  ASUB
    ELSEIF (C.EQ.’X') THEN
    CALL XSUB
    ELSE
    CALL OTHER
    ENDIF
    . . . . . .
```

Оператор ENDIF.

Оканчивает оператор блока IF

Синтаксис:

ENDIF

***Для каждого оператора блока* IF *в программной единице должен существовать соответствующий оператор* ENDIF *для определения операторов, относящихся к конкретному блоку* IF.**

Пример:

```
C   Пример оператора ENDIF
        I = 0
    IF (I .LT. 0) THEN
    X=-1
    Y=-1
    ENDIF
    . . . . . .
```

Концепция уровня блока IF

- **IF уровень** определяет правило вложения для блока **IF** и связанных с ним операторов и определяет зону влияния **IF**, **ELSEIF** и **ELSE** блоков.
- Для того чтобы не совершать алгоритмических ошибок при создании программ на языке Фортран, нужно четко представлять себе концепцию уровней блоков **IF**.

Концепция уровня блока **IF** *и связанных с ним операторов следующая:*

Для каждого оператора его IF уровнем является n1-n2.

Где

n1 - ***число блоков*** **IF** ***операторов, от начала программной единицы, в которой используется данный оператор, включая текущий.***

n2 - ***число*** **ENDIF** ***операторов от начало программной единицы, до данного, исключая его.***

КОНЦЕПЦИЯ УРОВНЯ БЛОКА IF

- …ень любого оператора должен быть больше или равен 0.
- IF-уровень блока **IF, ELSEIF, ELSE, ENDIF** должен быть больше 0.
- IF-уровень каждого последнего оператора должен быть **0**.

КОНЦЕПЦИЯ УРОВНЯ БЛОКА IF.

Пример:

```
C   Простейший блок IF, который перескакивает
C   группу операторов, если выражение FALSE
C
      IF(I.LT.10) THEN
C
C   . Набор операторов, вычисляемых
C    только если I.LT.10
C
      ENDIF
```

Пример:

```
C   Блок IF с операторами ELSEIF
    IF (J .GT. 1000) THEN
C   .. Набор операторов, вычисляемых,
C      только если J.GT.1000
    ELSEIF (J .GT. 100) THEN
C   .. Набор операторов, вычисляемых,
C      только если J.GT.100 и J.LE.1000
    ELSEIF (J .GT. 10) THEN
C   .. Набор операторов, вычисляемых,
C      только если J.GT.10 и J.LE.100
    ELSE
C   .. Набор операторов, вычисляемых,
C      только если J.LE.10
    ENDIF
```

Пример:

```
C   Вложенная конструкция IF без использования ELSEIF
    IF(I .LT. 100) THEN
C       Набор операторов, выполняемых, только если I.LT.100
    IF(J.LT.10)THEN
C       Набор операторов, выполняемых,
C       только если I.LT.100 и J.LT.10
    ENDIF
C       Набор операторов, выполняемых, только если I.LT.100
    ELSE
C       Набор операторов, выполняемых, только если I.GE.100
    IF(J.LT.10)THEN
C       Набор операторов, выполняемых,
C       только если I.GE.100 и J.LT.10
    ENDIF
C       Набор операторов, выполняемых, только если I.GE.100
    ENDIF
```

Оператор цикла

- При программировании циклических вычислительных алгоритмов могут быть использованы условные операторы.
- Для этих же целей служит специальный оператор цикла (или оператор **DO**), являющийся наиболее сложным и мощным из числа

Оператор DO.

Организует циклическое выполнение операторов, следующих за* DO *вплоть до оператора с меткой* slabel *включительно.

Синтаксис:

DO slabel [,]variable=expr1,expr2[,expr3]

Где

slabel - *операторная метка выполняемого оператора.*

variable - *целая переменная.*

expr1,expr2,expr3 - *целое выражение*

Особенности:

- **GO TO**, *присваиваемым* **GO TO**, *арифметическим* **IF**, *блоковым* **IF**, **ELSEIF**, **ELSE**, **ENDIF**, **RETURN**, **STOP**, **END**, *или оператором* **DO**.

- *исключая те, которые не запускаются внутри логического* **IF**.

ОПЕРАТОР ЦИКЛА

Ограничения на выполнение оператора DO:

- IF
- ELSE IF
- ELSE
- END DO

Параметр цикла оператора DO не может быть изменен каким-либо образом посредством операторов, содержащихся внутри области действия!!!

Вход в область выполнения цикла извне не допускается!!!

(специальные особенности позволяют войти в область цикла извне см. описание метакоманды $DO66).

Пример:

```
C     Высвечивание номеров с 1 до 11 на экране
C     здесь показана конечная величина переменной цикла
      DO 200 I=1,10
200       WRITE(*,'(I5)') I
          WRITE(*,'(I5)') I
C     Заполнение 20-элементного действительного массива
      DIMENSION  ARRAY(20)
      DO 1,I = 1,20
1     ARRAY(I) = 0.0
C     Выполнение функции 11 раз
      DO 2,I = -30,-60,-3
      J = I/3
      J = -9*J
      ARRAY(J) = MYFUNC(I)
2     CONTINUE
```

Оператор продолжения CONTINUE.

***Оператор* CONTINUE *преимущественно используется как удобная точка для размещения метки, в частности - как конечный оператор в операторе цикла* DO.**

Синтаксис:

CONTINUE

Использование* CONTINUE *не влияет на эфективность программы.

Пример:

```
C    Пример оператора CONTINUE
C
     DO 10,I=1,10
     IARRAY(I)=0
10   CONTINUE
```

ОПЕРАТОР ПРОДОЛЖЕНИЯ CONTINUE

Операторы останова и окончания

- Операторы останова и окончания:

  PAUSE, STOP и **END**.

Оператор PAUSE.

***Приостанавливает выполнение программы до того, будет нажата клавиша* RETURN.**

Синтаксис:

PAUSE [n]

Где

n - *это символьная константа, либо строка не более чем из пяти цифр.*

Особенности:

- *Оператор* **PAUSE** *приостанавливает выполнение программы до команды продолжать.*
- *Параметр* **n**, *если он есть, выдается на экран как приглашение, требующее ввода с клавиатуры.*
- *Если* **n** *нет, на экран выдается такое сообщение :*

PAUSE. Please press <return> to continue

- *После нажатия на клавишу Ввод (ENTER) выполнение программы возобновится, как если бы был выполнен оператор* **CONTINUE**.

ОПЕРАТОР PAUSE

Пример:

```
C   Пример оператора PAUSE
C
      IF (INARN .EQ. 0) GO TO 300
      PAUSE 'WARNING : INARM IS NONZERO'
300   CONTINUE
```

Оператор STOP.

Оператор* STOP *оканчивает программу.

Синтаксис:

STOP [n]

Где

n - *это символьная константа, либо строка не более чем из пяти цифр.*

Особенности:

- *Параметр* **n**, *если он есть, высвечивается на экран, когда программа оканчивается.*
- *Если* **n** *нет, на экран выдается такое сообщение :*

STOP – Program terminated

Пример:

```
C   Пример оператора STOP
C
      IF (IERROR .EQ. 0) GO TO 200
      STOP 'Определена ошибка'
200   CONTINUE
```

Оператор END.

В подпрограмме обладает тем же действием, что и оператор RETURN.

В главной программе оканчивает выполнение программы.

Синтаксис:

END

Особенности:

- *Оператором* **END** *должна заканчиваться каждая программная единица.*
- *Оператор* **END** *должен стоять на отдельной строке и в единственном виде (без других операторов), а также без метки.*
- *Не допускается продолжение строки, на которой находится* **END.**
- *Не допускается расположение на строке, где стоит* **END**, *никаких других операторов, в том числе* **RETURN** *и* **ENDIF.**

Пример:

```
C   Пример оператора END
C
C   оператор END должен быть
C
C   последним оператором в программе
C
      PROGRAM MYPROG
      WRITE (*,'(10H HI WORLD!)')
      END
```